The IEEE Verilog 1364-2000 Standard

What's New, and Why You Need It

by Stuart Sutherland

Sutherland HDL, Inc.

Verilog Training and Consulting Experts

Presented at the HDLCON-2000 Conference
March 10, 2000 San Jose, California

Verilog-2000 Update

- The specification of the Verilog-2000 standard is complete
  - Final draft completed March 1st, 2000
  - The final IEEE balloting process has started
  - Expect Verilog-2000 to be ratified in Q3-2000
  - The official standard will be IEEE Std. 1364-2000

© 2000, Sutherland HDL, Inc., www.sutherland-hdl.com

Why a New Standard?

- Add enhancements to Verilog
  - Design methodologies are evolving
    - System level design, intellectual property models, design re-use, very deep submicron, etc.
  - Cliff Cumming's "Top Five Enhancement Requests" from HDLCON-1996
- Clarify ambiguities in Verilog 1364-1995
  - The 1364-1995 reference manual came the Gateway Design Automation Verilog-XL User's Manual
  - Verilog-2000 more clearly defines Verilog syntax and semantics

© 2000, Sutherland HDL, Inc., www.sutherland-hdl.com

Goals for Verilog-2000

- Enhance Verilog for
  - Higher level, abstract system level modeling
  - Intellectual Property (IP) modeling
  - Greater timing accuracy for very deep submicron
- Make Verilog even easier to use
- Correct errata and ambiguities
- Maintain backward compatibility — existing models will work with the new standard
- Ensure that EDA vendors will implement all enhancements!

© 2000, Sutherland HDL, Inc., www.sutherland-hdl.com

The IEEE 1364 Verilog Standards Committee

- A main working group
  - Final approval of all changes to 1364-1995
  - About 20 active participants
- Three task forces
  - Behavioral Task Force (Cliff Cummings, chair)
    - RTL and behavioral modeling enhancements
  - ASIC Task Force (Steve Wadsworth, chair)
    - ASIC and FPGA library modeling enhancements
  - PLI Task Force (Drew Lynch, Stu Sutherland, co-chairs)
    - PLI enhancements

© 2000, Sutherland HDL, Inc., www.sutherland-hdl.com

Overview of HDL Enhancements

- 33 major enhancements were added to the Verilog HDL
  - Brief description and examples
  - New reserved words
- Errata and clarifications
  - Dozens of corrections were made to 1364-1995
  - Do not affect Verilog users
  - Very important to Verilog tool implementors
  - Not listed in this paper — refer to the 1364-2000 Verilog Language Reference Manual (LRM)

© 2000, Sutherland HDL, Inc., www.sutherland-hdl.com

1: Verilog Configurations

- Verilog-1995 leaves design management up to the software tools
  - Every tool has different ways to manage large designs
- Verilog-2000 adds configuration blocks
  - All software tools will have a consistent method
  - The version for each module instance can be specified
    - Virtual libraries specified within Verilog source code
    - Physical file locations specified in a “map” file
  - New reserved words added: ***config***, ***endconfig***, ***design***, ***instance***, ***cell***, ***use***, ***liblist***

© 2000, Sutherland HDL, Inc., www.sutherland-hdl.com

Verilog Configuration Notes

- Verilog design hierarchy is modeled the same as always
- Configurations can be used to specify which module source code should be used for each instance of a module.
  - With Verilog-1995, it is up to the simulator on how to specify which model version should be used for each instance (if the simulator can do it at all)
- The configuration block is specified outside of all modules
  - Can be in the same file as the Verilog source code
  - Can be in a separate file
  - Verilog model source code does not need to be modified in order to change the design configuration!
- A separate file maps logical library names to physical file locations
  - Verilog source code does not need to be modified when a design is moved to a different physical source location!

© 2000, Sutherland HDL, Inc., www.sutherland-hdl.com

Verilog Configuration Example

Verilog Design

```
module test;
  ...
  myChip dut (...);
  ...
endmodule
```

```
module myChip(...);
  ...
  adder a1 (...);
  adder a2 (...);
  ...
endmodule
```

Configuration Block (part of Verilog source code)

```
/* define a name for this configuration */
config cfg4

  /* specify where to find top level modules */
  design rtlLib.top

  /* set the default search order for finding
     instantiated modules */
  default liblist rtlLib gateLib;

  /* explicitly specify which library to use
     for the following module instance */
  instance test.dut.a2 liblist gateLib;
endconfig
```

Library Map File

```
/* location of RTL models (current directory) */
library rtlLib "./*.v";

/* Location of synthesized models */
library gateLib "./synth_out/*.v";
```


© 2000, Sutherland HDL, Inc., www.sutherland-hdl.com

2: Verilog Generate

- Verilog-2000 adds true generate capability
  - Use ***for*** loops to generate any number of instances of:
    - Modules, primitives, procedures, continuous assignments, tasks, functions, variables, nets
  - Use ***if–else*** and ***case*** decisions to control what instances are generated
    - Provides greater control than the VHDL generate
  - New reserved words added:
    - ***generate***, ***endgenerate***, ***genvar***, ***localparam***

© 2000, Sutherland HDL, Inc., www.sutherland-hdl.com

Verilog Generate Example

```
module multiplier (a, b, product);
  parameter a_width = 8, b_width = 8;
  localparam product_width = a_width + b_width;
  input   [a_width-1:0]       a;
  input   [b_width-1:0]       b;
  output  [product_width-1:0] product;

  generate
    if ((a_width < 8) || (b_width < 8))
      CLA_multiplier #(a_width, b_width) u1 (a, b, product);
    else
      WALLACE_multiplier #(a_width, b_width) u1 (a, b, product);
  endgenerate

endmodule
```

localparams are constants, which cannot be redefined

- **If the input bus widths are 8-bits or less, generate and instance of a carry-look-ahead multiplier**
- **If the input bus widths are greater than 8-bits, generate an instance of a wallace-tree multiplier**

© 2000, Sutherland HDL, Inc., www.sutherland-hdl.com

3: Constant Functions

- Verilog-2000 adds constant functions
  - Same syntax as standard Verilog functions
  - Limited to statements that can be evaluated at compile time
  - Can be called anywhere a constant expression is required
    - Vector width declarations
    - Array declarations
    - Replicate operations
- Provides for more scalable, re-usable models

© 2000, Sutherland HDL, Inc., www.sutherland-hdl.com

Constant Functions Example

```
module ram (...);
  parameter RAM_SIZE = 1024;
  parameter ADDRESS = 12;
  input [ADDRESS-1:0] address_bus;
```

Verilog 1995:

Vector widths can be calculated using simple constant expressions

```
module ram (...);
  parameter RAM_SIZE = 1024;
  input [clogb2(RAM_SIZE)-1:0] address_bus;
  ...
  function integer clogb2;
    input depth;
    integer i;
    begin
      clogb2 = 1;
      for (i = 0; 2**i < depth; i = i + 1)
        clogb2 = i + 1;
    end
  endfunction
  ...
```

Verilog 2000:

Vector widths can be calculated using complex constant functions

© 2000, Sutherland HDL, Inc., www.sutherland-hdl.com

4: Indexed Vector Part Selects

- Verilog-2000 adds the capability to use variables to select a group of bits from a vector
  - The starting point of the part-select can vary
  - The width of the part-select remains constant

```
reg [63:0] word;
reg  [3:0] byte_num; //a value from 0 to 7
wire [7:0] byteN = word[byte_num*8 +: 8];
```

The starting point of the part-select is variable

The width of the part-select is constant

+: indicates the part-select increases from the starting point
-: indicates the part-select decreases from the starting point

© 2000, Sutherland HDL, Inc., www.sutherland-hdl.com

5: Multi-dimensional Arrays

- Verilog-1995 allows 1-dimensional arrays of reg, integer and time variables
  - Typically used to model RAM and ROM memories
- Verilog-2000 adds:
  - Multidimensional arrays of any variable data type
  - Multidimensional arrays of any net data type

```
//declare a 3-dimensional array of 8-bit wire nets
wire [7:0] array3 [0:255][0:255][0:15];

//select one word out of a 3-dimensional array
wire [7:0] out3 = array3[addr1][addr2][addr3];
```


© 2000, Sutherland HDL, Inc., www.sutherland-hdl.com

6: Array Bit and Part Selects

- Verilog-2000 adds:
  - Bit-selects out of an array
  - Part-selects out of an array

```
//select the high-order byte of one word in a
//2-dimensional array of 32-bit reg variables
reg [31:0] array2 [0:255][0:15];
wire [7:0] out2 = array2[100][7][31:24];
```


© 2000, Sutherland HDL, Inc., www.sutherland-hdl.com

7: Signed Arithmetic Extensions

- Verilog-2000 adds:
  - reg and net data types can be declared as signed

```
reg  signed [63:0] data;
wire signed [11:0] address;
```

  - Function returns can be declared as signed

```
function signed [128:0] alu;
```

  - Literal integer numbers can be declared as signed

```
16'shC501  //a signed 16-bit hex value
```

  - New arithmetic shift operators, **<<<** and **>>>**, maintain the sign of a value
  - New **$signed()** and **$unsigned()** system functions can “cast” a value to signed or unsigned

© 2000, Sutherland HDL, Inc., www.sutherland-hdl.com

8: Power Operator

- Verilog-2000 add an exponential power operator
  - Represented by the ** token
  - Works like the C pow() function
  - If either operand is real, a real value is returned
  - If both operands are integers, an integer value is returned

```
module ram (...);
  parameter RAM_SIZE = 1024;
  input [(2**RAM_SIZE)-1:0] address_bus;
  ...
```


© 2000, Sutherland HDL, Inc., www.sutherland-hdl.com

9: Re-entrant Tasks and Recursive Functions

- Verilog-2000 adds automatic tasks and functions
  - Each call to the task/function allocates unique storage
    - In Verilog-1995, tasks and functions are static; each call shares the same storage space
  - Concurrent task calls will not interfere with each other
  - Recursive calls to a function are stacked
  - New reserved word added: ***automatic***

```
function automatic [63:0] factorial;
  input [31:0] n;
  if (n == 1)
    factorial = 1;
  else
    factorial = n * factorial(n-1);
endfunction
```

Recursive function call

© 2000, Sutherland HDL, Inc., www.sutherland-hdl.com

10: Comma-separated Sensitivity List

- Verilog-2000 adds a second syntax style for listing signals in a sensitivity list
  - Signals in the list can be separated with a comma
    - The old “or” separated list will still work

Verilog-1995

```
always @(sel or a or b or c or d)
  case (sel)
    2'b00: y = a;
    2'b01: y = b;
    2'b10: y = c;
    2'b11: y = d;
  endcase
```

Verilog-2000

```
always @(sel, a, b, c, d)
  case (sel)
    2'b00: y = a;
    2'b01: y = b;
    2'b10: y = c;
    2'b11: y = d;
  endcase
```


© 2000, Sutherland HDL, Inc., www.sutherland-hdl.com

11: Combinational Logic Sensitivity

- Verilog-2000 adds a “wildcard” token to indicate a combinational logic sensitivity list
  - The @* token indicates that an always procedure is automatically sensitive to any change on any input to that procedure

Verilog-1995

```
always @(sel or a or b or c or d)
  case (sel)
    2'b00: y = a;
    2'b01: y = b;
    2'b10: y = c;
    2'b11: y = d;
  endcase
```

Verilog-2000

```
always @*
  case (sel)
    2'b00: y = a;
    2'b01: y = b;
    2'b10: y = c;
    2'b11: y = d;
  endcase
```


© 2000, Sutherland HDL, Inc., www.sutherland-hdl.com

12: Enhanced File I/O

- Verilog-1995 has limited built-in file I/O tasks
  - Up to 31 files can be opened for writing
    - Only ASCII characters can be written to files
  - More complex file I/O is done using the Verilog Programming Language Interface (PLI)
- Verilog-2000 adds:
  - The ability to open up to 2^{30} files
  - New, built-in file I/O tasks: **$ferror**, **$fgetc**, **$fgets**, **$fflush**, **$fread**, **$fscanf**, **$fseek**, **$fsscanf**, **$ftel**, **$rewind**, **$sformat**, **$swrite**, **$swriteb**, **$swriteh**, **$swriteo**, **$ungetc**

© 2000, Sutherland HDL, Inc., www.sutherland-hdl.com

13: Automatic Width Extension Past 32 bits

- In Verilog-1995:
  - Verilog assignments zero fill when the left-hand side is wider than the right-hand side
  - Unsized integers default to 32-bits wide; therefore, the widths of integers must be hard-coded

Verilog-1995

```
parameter WIDTH = 64;
reg [WIDTH-1:0] data;
data = 'bz;    //fills with 'h00000000zzzzzzzz
data = 64'bz;  //fills with 'hzzzzzzzzzzzzzzzz
```

- Verilog-2000 will automatically extend a logic Z or X to the full width of the left-hand side

Verilog-2000

```
parameter WIDTH = 64;
reg [WIDTH-1:0] data;
data = 'bz;    //fills with 'hzzzzzzzzzzzzzzzz
```

14: Default Nets with Continuous Assigns

- Verilog-2000 will default to a net data type on the left-hand side of any continuous assignment
  - The vector width is the size of the right-hand side expression, if not connected to a port of the module
  - In Verilog-1995, the left-hand side must be explicitly declared, if not connected to a port of the module

Verilog-1995

```
module mult32 (y, a, b);
  output [63:0] y;
  input  [31:0] a, b;
  assign y = a * b;  //defaults to wire, width of port y
  assign n = a * b;  //ERROR: 'n' not declared
endmodule
```

Verilog-2000

```
  assign n = a * b;  //defaults to wire, width of (a * b)
```


© 2000, Sutherland HDL, Inc., www.sutherland-hdl.com

15: Disable Default Net Declarations

- In Verilog-1995, undeclared signals can default to a wire data type
  - The default data type can be changed to another net data type using `default_nettype <data_type>
- Verilog-2000 provides a means to disable default net declarations
  - `default_nettype **none**
  - Any undeclared signals will be a syntax error
    - Prevents hard-to-debug wiring errors due to a mistyped name
  - none is not a new reserved word

© 2000, Sutherland HDL, Inc., www.sutherland-hdl.com

16: Explicit In-line Parameter Passing

- Verilog-2000 adds the ability to explicitly name parameters when passing parameter values
  - Provides better self-documenting code
  - Parameter values can be passed in any order

```
module ram (...);
  parameter WIDTH = 8;
  parameter SIZE = 256;
  ...
endmodule
```

Verilog-1995

```
module my_chip (...);
  ...
  RAM #(8,1023) ram2 (...);
endmodule
```

Verilog-2000

```
module my_chip (...);
  ...
  RAM #(.SIZE(1023)) ram2 (...);
endmodule
```


© 2000, Sutherland HDL, Inc., www.sutherland-hdl.com

17: Combined Port/Data Type Declarations

- Verilog-2000 permits combining port declarations and data type declarations into one statement

Verilog-1995

```
module mux8 (y, a, b, en);
  output [7:0] y;
  input  [7:0] a, b;
  input        en;

  reg  [7:0] y;
  wire [7:0] a, b;
  wire       en;
  ...
```

Verilog-2000

```
module mux8 (y, a, b, en);
  output reg  [7:0] y;
  input  wire [7:0] a, b;
  input  wire       en;
  ...
```


© 2000, Sutherland HDL, Inc., www.sutherland-hdl.com

18: ANSI-style Port Lists

- Verilog-2000 adds ANSI C style input and output declarations
  - For modules, tasks and functions

Verilog-1995

```
module mux8 (y, a, b, en);
  output [7:0] y;
  input  [7:0] a, b;
  input        en;

  reg  [7:0] y;
  wire [7:0] a, b;
  wire       en;
  ...
```

Verilog-2000

```
module mux8 (output reg  [7:0] y,
             input  wire [7:0] a,
             input  wire [7:0] b,
             input  wire       en);
  ...
```


© 2000, Sutherland HDL, Inc., www.sutherland-hdl.com

19: Reg Declaration With Initialization

- Verilog-2000 permits initializing variables at the time they are declared
  - The initialization is executed in time-step zero, just like initial procedures

Verilog-1995

```
reg clock;

initial
  clk = 0;
```

Verilog-2000

```
reg clock = 0;
```


© 2000, Sutherland HDL, Inc., www.sutherland-hdl.com

20: "Register" Changed To "Variable"

- The Verilog-2000 standard changes the term "register" to "variable"
  - "register" is not a reserved word; it is just a term
  - Since its inception in 1984, Verilog manuals have used the term "register" to describe a class of data types
    - reg (unsigned variable), integer (signed variable), real (double precision variable), etc.
  - The term "register" often confuses new Verilog users
    - register is a hardware term for storage elements
    - Verilog registers do not imply a hardware register

© 2000, Sutherland HDL, Inc., www.sutherland-hdl.com

21: Enhanced Conditional Compilation

- Verilog-1995 supports limited conditional compilation
  - The `ifdef, `else and `endif compiler directives
- Verilog-2000 adds more extensive conditional compilation control
  - New directives: **`ifndef**, **`elsif** and **`undef**

© 2000, Sutherland HDL, Inc., www.sutherland-hdl.com

22: File and Line Compiler Directives

- Verilog-2000 adds file and line compiler directives
  - New directives: **`file** and **`line**
  - Document the original location of Verilog source code
    - Verilog tools often include file name and line number information in error and warning messages
    - If a pre-process utility program modifies the Verilog source code, the original file and line information could be lost

© 2000, Sutherland HDL, Inc., www.sutherland-hdl.com

23: Attributes

- Verilog-2000 adds “attribute” properties
  - A standard means to specify non-Verilog tool specific information to Verilog models
  - Adds new tokens **(*** and ***)**
  - Eliminates need to hide commands in comments
  - The standard does not define any specific attributes
    - Software vendors can define proprietary attributes
    - Other standards might define standard attributes

Verilog-1995

```
case (1'b1) /* synopsys parallel_case */ //1-hot FSM
```

Verilog-2000

```
(* parallel case *) case (1'b1)  //1-hot FSM
```


© 2000, Sutherland HDL, Inc., www.sutherland-hdl.com

24: Standard Random Number Generator

- Verilog-2000 defines the C source code for the generator used by $random
  - All simulators can generate the same random number sequence when given the same seed value
    - Simulation results from different simulators can be compared
    - New products do not need to re-invent the wheel
  - Uses the random number generator from Verilog-XL

© 2000, Sutherland HDL, Inc., www.sutherland-hdl.com

25: Enhanced Invocation Option Tests

- Verilog-1995 contains a true/false test to see if simulation was invoked with a specific option
  - $test$plusargs
- Verilog-2000 adds the ability to read arguments of invocation options
  - New system function: **$value$plusargs**

© 2000, Sutherland HDL, Inc., www.sutherland-hdl.com

26: Enhanced PLA Modeling

- Verilog-2000 extends the capability of the PLA system tasks ($async$or$array, $async$and$array, etc.)
  - In Verilog-1995, arguments had to be scalar
  - In Verilog-2000, arguments can be vectors

© 2000, Sutherland HDL, Inc., www.sutherland-hdl.com

27: Accurate BNF, with Subsections

- The Verilog-1995 BNF (Backus-Naur Form) had errors and inconsistencies
- Verilog-2000 contains a much stronger BNF definition of the Verilog language
  - Consistent terminology
  - More definitive terms
  - Divided into sub-sections to make it easier to find specific definitions
  - Checked for accuracy

© 2000, Sutherland HDL, Inc., www.sutherland-hdl.com

28: On-detect Pulse Error Propagation

- Verilog-1995 has on-event pulse error propagation
  - A pulse is a glitch on the inputs of a module path that is less than the delay of the path
  - An input pulse propagates to a path output as an X, with the same delay as if a valid input change had propagated to the output
- Verilog-2000 adds on-event pulse error propagation
  - As soon as an input pulse is detected, a logic X is propagated to a path output, without the path delay
  - New reserved words added:
    - **pulsestyle_onevent**, **pulsestyle_ondetect**

© 2000, Sutherland HDL, Inc., www.sutherland-hdl.com

29: Negative Pulse Detection

- Verilog-2000 adds negative pulse detections
  - Due to different rising-transition and falling-transition delays, it is possible for the trailing edge of a glitch to propagate before the leading edge has propagated
    - In Verilog-1995, a negative pulse is cancelled
  - Negative pulse detection will propagate a logic X for the duration of the negative pulse
  - New reserved words added:
    - **showcancelled**, **noshowcancelled**

© 2000, Sutherland HDL, Inc., www.sutherland-hdl.com

30: New Timing Constraint Checks

- Verilog-2000 adds new timing constraint checks
  - More accurately model very deep submicron input constraints:
  - New timing constraint tasks added:
    - **$removal**
    - **$recrem**
    - **$timeskew**
    - **$fullskew**
  - Refer to the proposed IEEE 1364-2000 Verilog standard for details on these tasks

© 2000, Sutherland HDL, Inc., www.sutherland-hdl.com

31: Negative Timing Constraints

- Verilog-2000 adds the ability to specify negative values for:
  - $setuphold setup and hold times
    - Adds new, optional arguments to the Verilog-1995 $setuphold task
  - $recrem recovery and removal times
    - A new timing check task in Verilog-2000

© 2000, Sutherland HDL, Inc., www.sutherland-hdl.com

32: Enhanced SDF support

- The Verilog-2000 standard defines:
  - How timing objects in SDF map to objects in Verilog
  - Based on the latest SDF standard, IEEE 1497-1999
- Verilog-2000 changes the syntax of the **specparam** constant
  - Can now be declared at the module level as well as within a specify block (to support SDF labels)
- Verilog-2000 adds a standard **$sdf_annotate** system task
  - Already a de-facto standard in all simulators

© 2000, Sutherland HDL, Inc., www.sutherland-hdl.com

33: Extended VCD Files

- Verilog-2000 adds new Value Change Dump (VCD) capabilities
  - Dump port change values
  - Dump strength level changes
  - Dump the time at which simulation finishes
  - New system tasks added:
    - **$dumpports**, **$dumpportsall**, **$dumpportsoff**, **$dumpportson**, **$dumpportslimit** and **$dumpportsflush**

© 2000, Sutherland HDL, Inc., www.sutherland-hdl.com

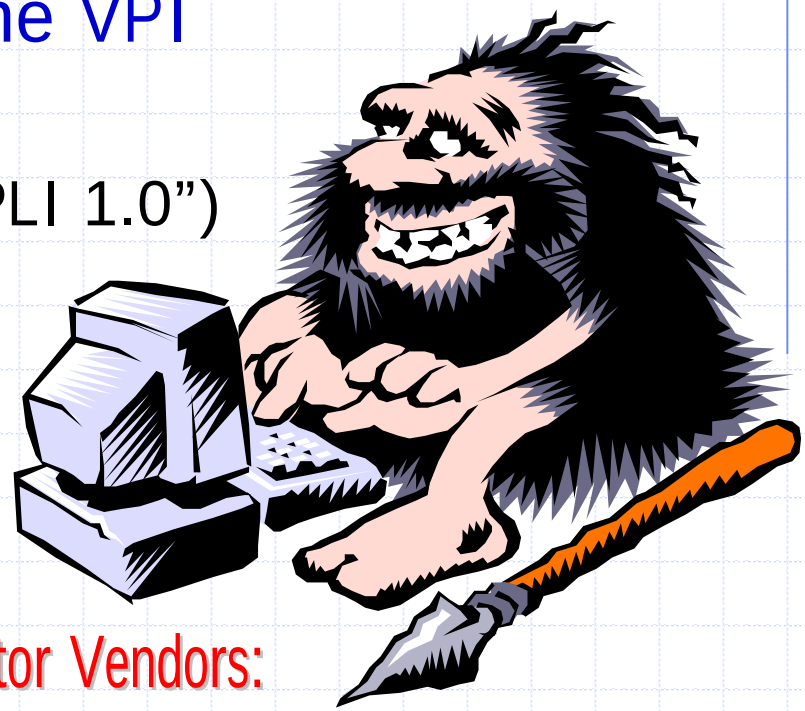
PLI Enhancements

- Several enhancements added to the VPI library
  - Simulation control
    - Stop, finish, save, restart, etc.
  - Support for new Verilog-2000 HDL constructs
    - Array of instances, attributes, signed arithmetic, recursive functions, enhanced file I/O, etc.
- Maintenance updates to TF and ACC libraries
  - Corrected errata
  - Clarified ambiguities

© 2000, Sutherland HDL, Inc., www.sutherland-hdl.com

The VPI Library Is The Future!

- All enhancements to the Verilog language will only be supported in the VPI library of the PLI
  - The TF and ACC libraries ("PLI 1.0") are only being maintained



Warning To Simulator Vendors:

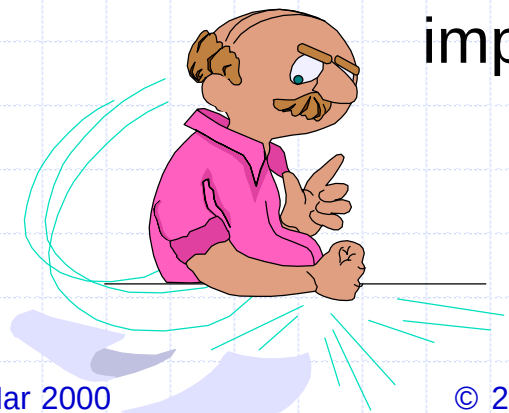
"PLI 1.0" is OVI's 1990 Verilog PLI standard; It isn't 1990 anymore!

Your customers do not want prehistoric simulators!

© 2000, Sutherland HDL, Inc., www.sutherland-hdl.com

When Will These Enhancements Be Available?

- The "official" word from several EDA vendors is:
  - They will not comment on future product plans
  - They will not begin to implement Verilog-2000 until it is ratified (expected to be mid-2000)
- The "unofficial" word from EDA vendors is:
  - Some have already started implementing Verilog-2000
  - One essentially says they do not see any need to implement the new features in Verilog-2000



Tell you simulator, synthesis and other Verilog tools providers that you want these Verilog-2000 enhancements **NOW!**

© 2000, Sutherland HDL, Inc., www.sutherland-hdl.com

Summary

- Verilog-2000 is complete
  - The proposed IEEE 1364-2000 Verilog standard is now in the final balloting phase
- Verilog-2000 contains
  - Over 30 major enhancements
  - Many clarifications and errata corrections
- Verilog-2000 adds powerful capabilities
  - Greater deep submicron accuracy
  - More abstract system level modeling
  - Scalable, re-usable modeling
- Final approval is expected in the summer of 2000

© 2000, Sutherland HDL, Inc., www.sutherland-hdl.com

About The Author

- Stuart Sutherland
  - President of Sutherland HDL, Inc., Portland, Oregon
  - Provides expert Verilog design consulting and training
  - More than 15 years design experience, and over 12 years working with Verilog
  - Author of “The Verilog HDL Quick Reference Guide”, “The Verilog PLI Quick Reference Guide” and “The Verilog PLI Handbook”, a 750 page reference book
  - Member of the IEEE 1364 Verilog standards committee since 1993
  - Co-chair of the PLI task force within the committee

© 2000, Sutherland HDL, Inc., www.sutherland-hdl.com